W9 Classifiers

- Clustering (unsupervised classification)
  - 1-D hypothesis testing
  - clustering with Gaussian Mixture models (GMM)
  - K-means clustering algorithm
  - hierarchical clustering algorithm
- (supervised) Classification
  - potpourri of supervised classification methods:

    naive Bayes, quadratic discriminant analysis, GMM, KNN, support vector machines
  - classification comparison with ROC curves

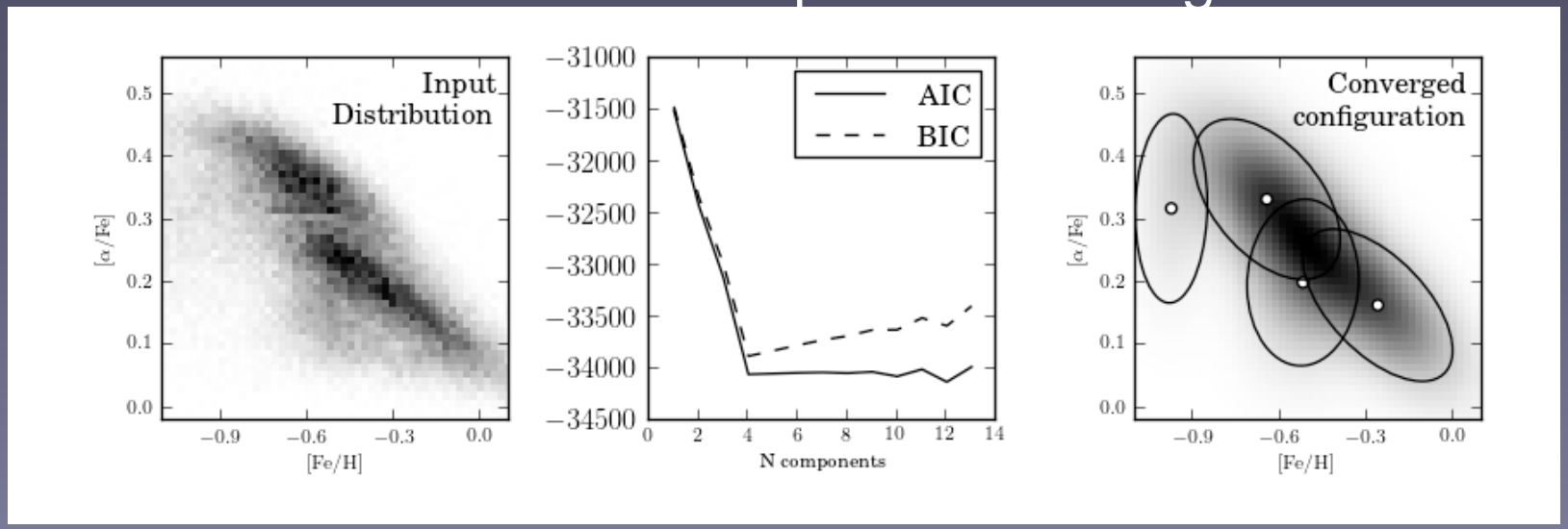
Clustering (unsupervised classification)

Here “unsupervised” means that there is no prior information about the number and properties of clusters.

“Clustering” in astronomy refers to a number of different aspects of data analysis.

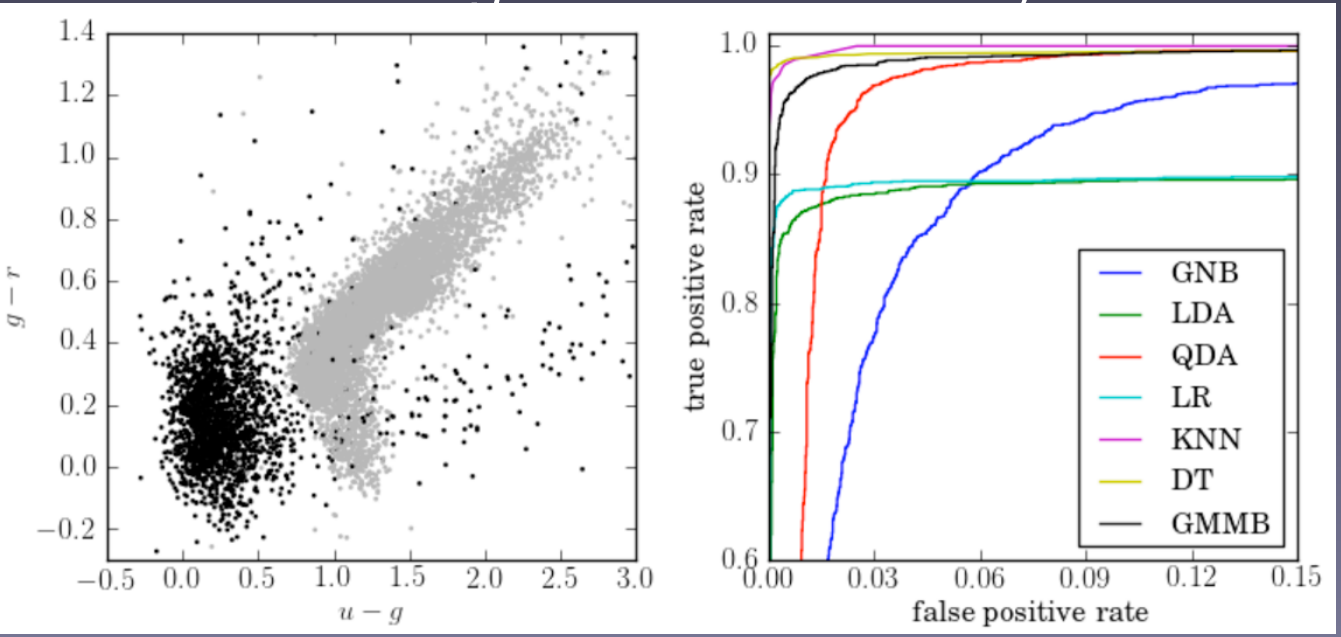
Given a multivariate point data set, we can ask whether it displays any structure, that is, concentrations of points. Alternatively, when a density estimate is available we can search for “overdensities.”

Another way to interpret clustering is to seek a partitioning or segmentation of data into smaller parts according to some criteria.



Supervised classification

Here “supervised” means that there is prior information about the number and properties of clusters: for a training sample, we know the so-called “class labels” (for each data point in the training sample, we know to which cluster it belongs; characterizing these known clusters is typically easier than finding and characterizing unknown clusters)



Supervised classification

There are two basic types of classification methods: **generative classification** methods model the underlying density field (i.e. it relies on density estimation methods), and **discriminative classification** methods which focus on finding the decision boundary which separates classes directly. The former are easier to interpret, the latter often work better in high-D cases.

Generative classification

Given a set of data $\{\mathbf{x}\}$ consisting of N points in D dimensions, such that x_i^j is the jth feature of the ith point, and a set of discrete labels $\{y\}$ drawn from K classes, with values y_k, Bayes' theorem describes the relation between the labels and features:

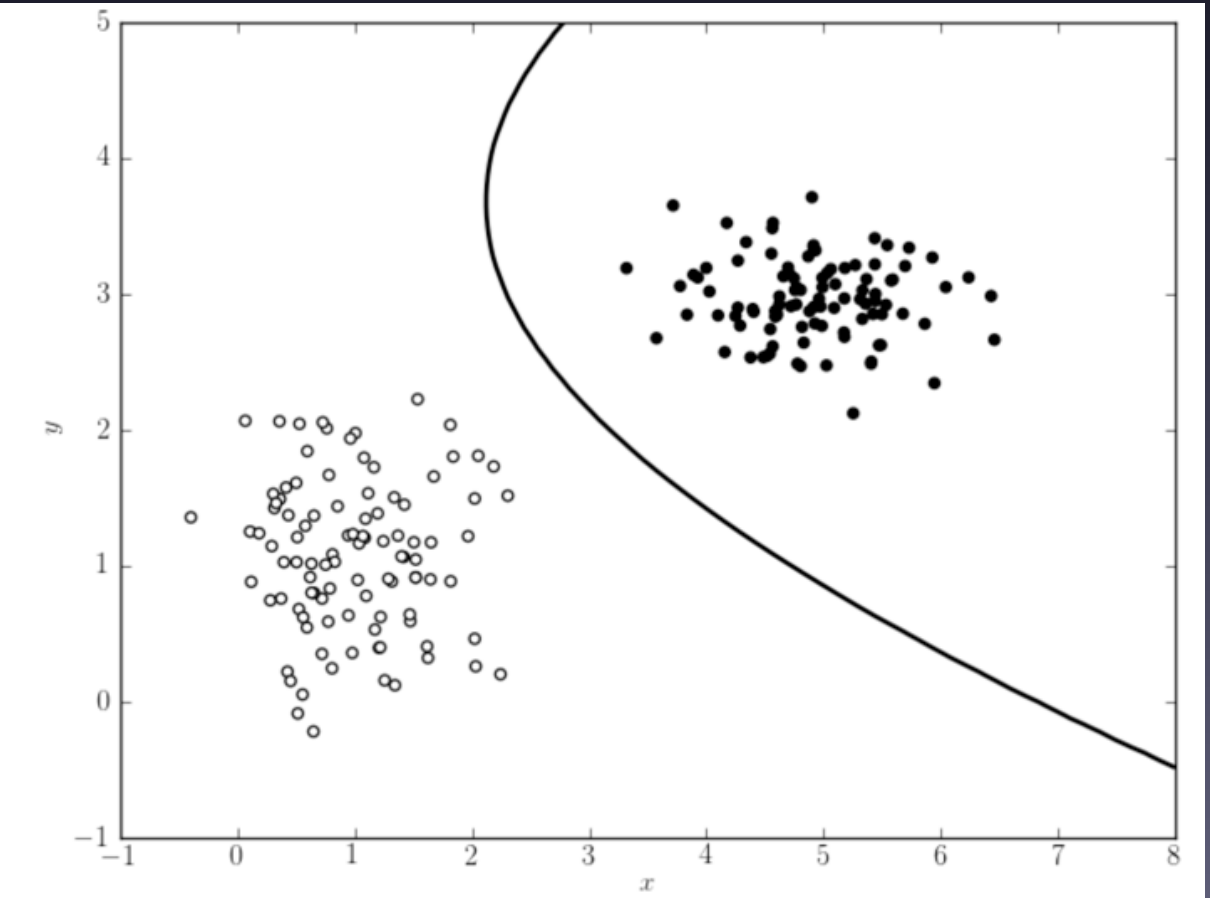
$$p(y_k|\mathbf{x}_i) = \frac{p(\mathbf{x}_i|y_k)p(y_k)}{\sum_i p(\mathbf{x}_i|y_k)p(y_k)}. \tag{9.6}$$

If we knew the full probability densities $p(\mathbf{x}, y)$ it would be straightforward to estimate the classification likelihoods directly from the data. If we chose not to fully sample $p(\mathbf{x}, y)$ with our training set we can still define the classifications by drawing from $p(y|\mathbf{x})$ and comparing the likelihood ratios between classes (in this way we can focus our labeling on the specific, and rare, classes of source rather than taking a brute-force random sample).

In generative classifiers we are modeling the class-conditional densities explicitly, which we can write as $p_k(\mathbf{x})$ for $p(\mathbf{x}|y = y_k)$, where the class variable is, say, $y_k = 0$ or $y_k = 1$. The quantity $p(y = y_k)$, or π_k for short, is the probability of any point having class k, regardless of which point

The task of learning the best classifier then becomes the task of estimating the p_k's. This approach means we will be doing multiple separate *density estimates* using many of the techniques introduced

Discriminative classification



$$\text{completeness} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}},$$

$$\text{contamination} = \frac{\text{false positives}}{\text{true positives} + \text{false positives}}$$

In this simple example, perfect separation is possible; generally not the case!

Decision boundary

The *decision boundary* between two classes is the set of x values at which each class is equally likely; that is,

$$\pi_1 p_1(\mathbf{x}) = \pi_2 p_2(\mathbf{x}); \tag{9.14}$$

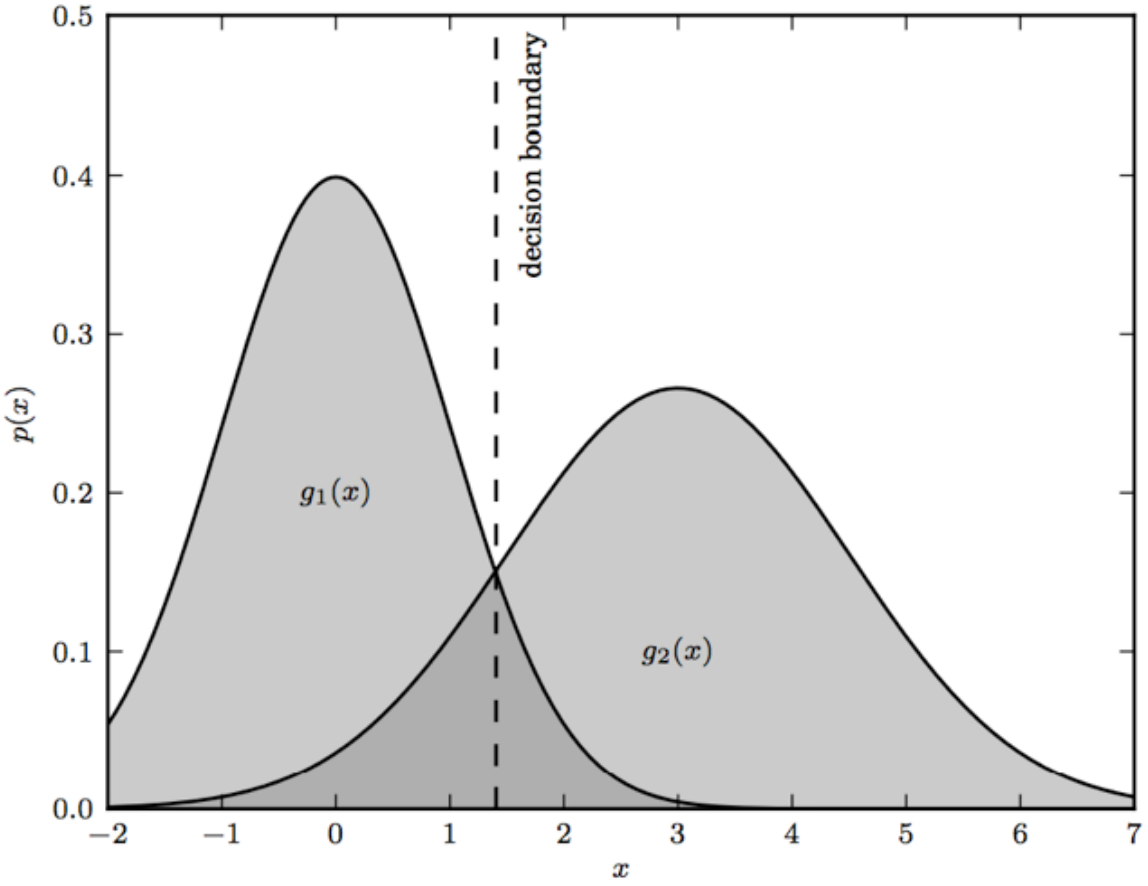
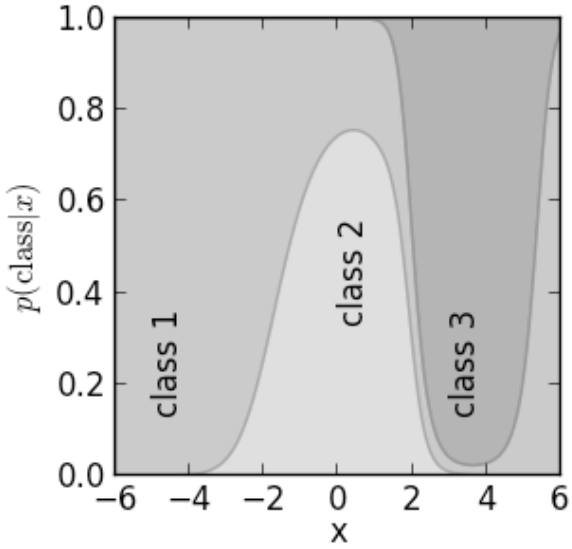
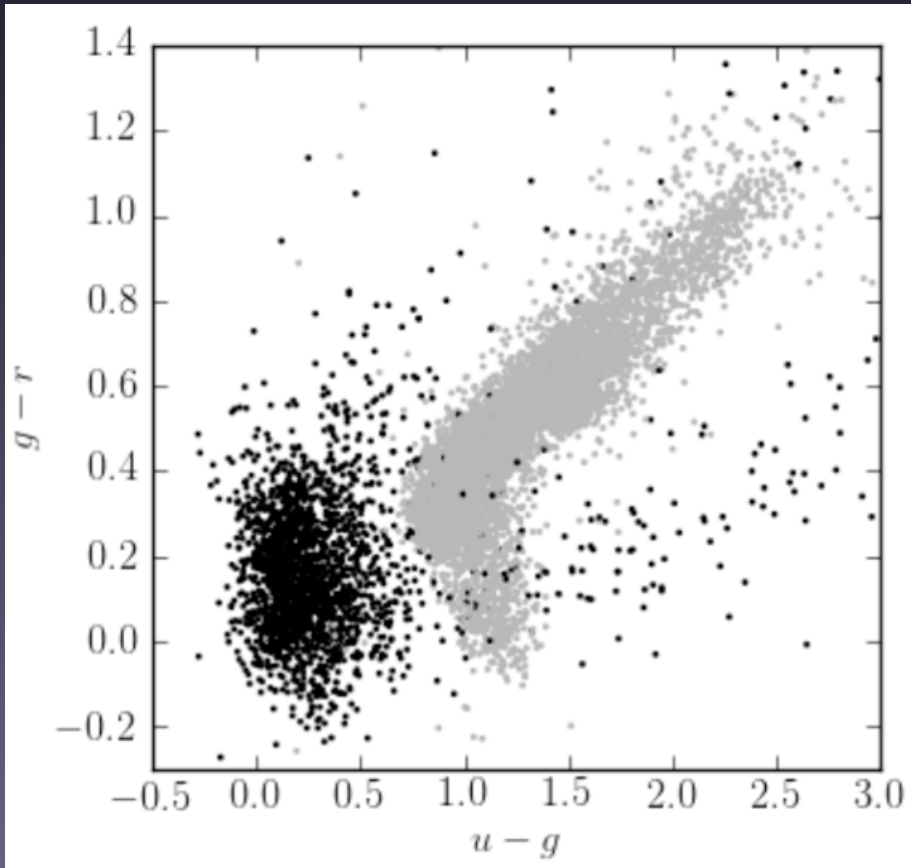


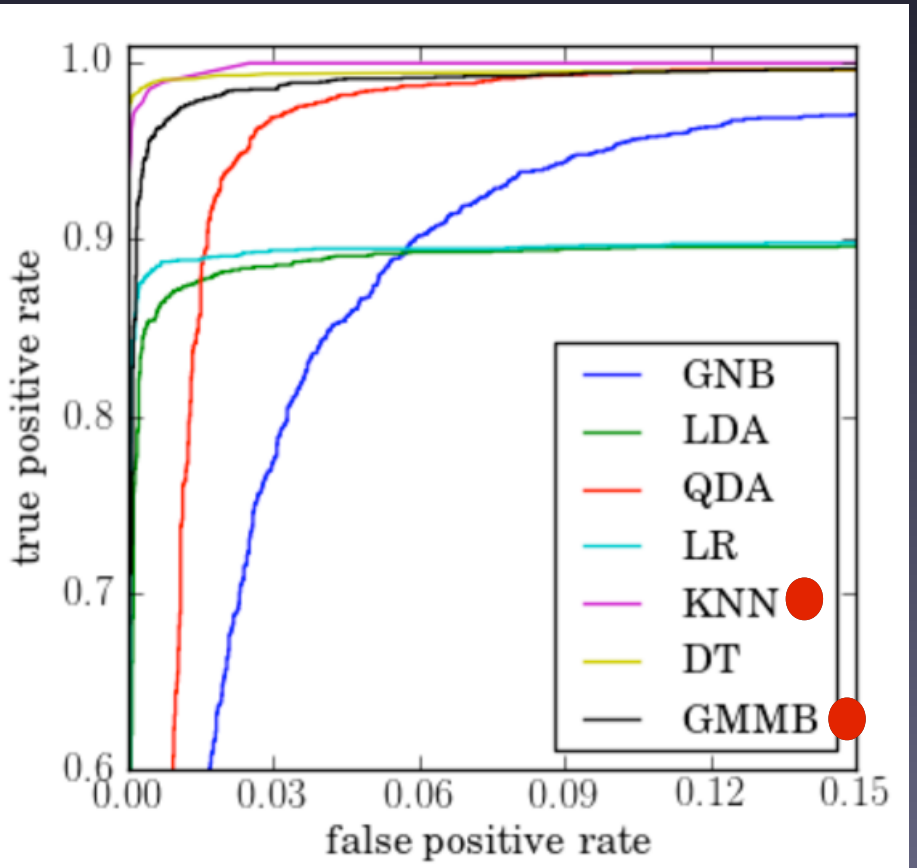
Figure 9.1.: An illustration of a decision boundary between two Gaussian distributions.



Comparison of classification methods with ROC curves

- an example of photometric star (gray) vs. quasar (black) separation using SDSS photometry (UV excess method)





Which method would you choose? Why? Would it fail for a different sample? There are many more methods in astroML, and even many many more in the literature...

TABLE 9.1.
Summary of the practical properties of different classifiers.

Method	Accuracy	Interpretability	Simplicity	Speed
Naive Bayes classifier	L	H	H	H
Mixture Bayes classifier	M	H	H	M
Kernel discriminant analysis	H	H	H	M
Neural networks	H	L	L	M
Logistic regression	L	M	H	M
Support vector machines: linear	L	M	M	M
Support vector machines: kernelized	H	L	L	L
K-nearest-neighbor	H	H	H	M
Decision trees	M	H	H	M
Random forests	H	M	M	M
Boosting	H	L	L	L

W8 Density estimation

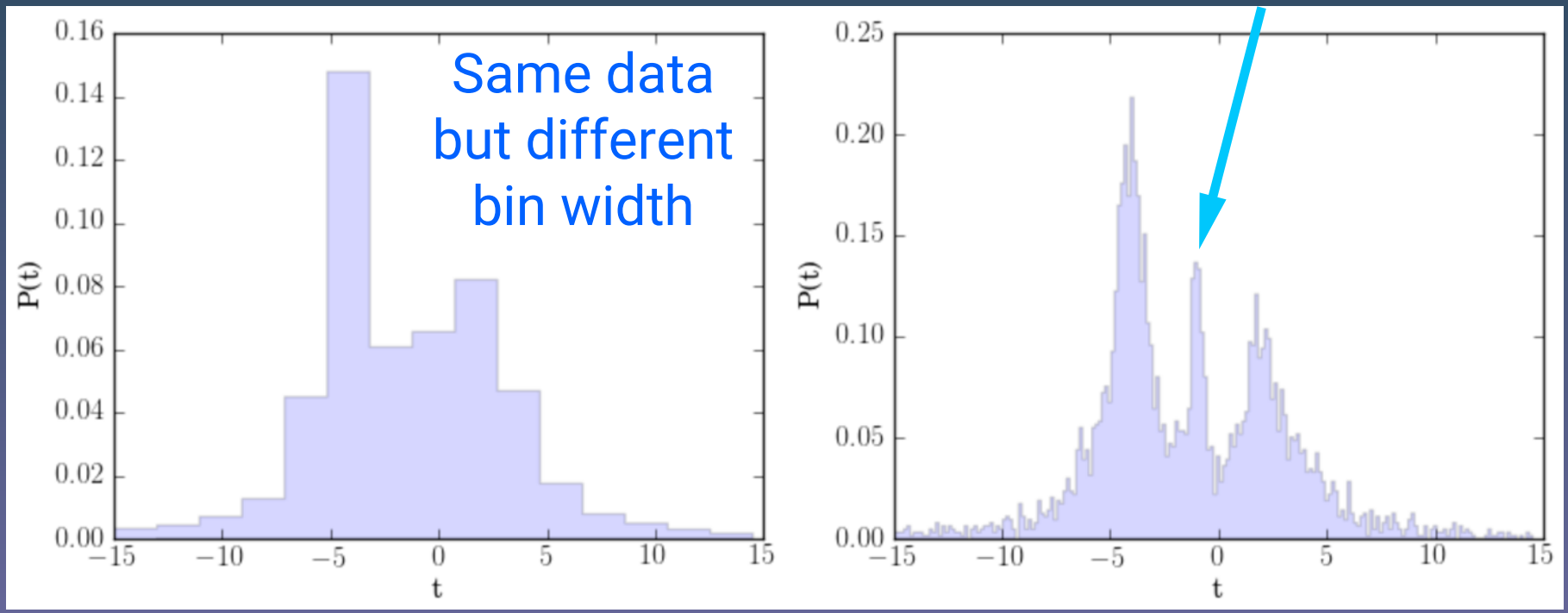
Density estimation is the act of estimating a continuous density field from a discretely sampled set of points drawn from that density field.

- One-dimensional introduction
  - Knuth's histograms
  - Scargle's Bayesian Blocks algorithm
  - Gaussian Mixture models
  - kernel density estimates (KDE)
  - the Wiener filter and connection to KDE
- Density estimation in high-D
  - high-D KDE
  - Bayesian nearest neighbor method
  - Extreme Deconvolution in high-D

What is a histogram?

- ---> Data modeled by a step function
- How do we determine/estimate/guess the bin width?
- Do all the bins have to have the same width?
- Do we really have to bin data to estimate model parameters?

Should we believe the middle peak?



Simple rules for estimating bin width

Various proposed methods for choosing optimal bin width typically suggest a value proportional to some estimate of the distribution's scale, and decreasing with the sample size. The most popular choice is "Scott's rule" which prescribes a bin width

$$\Delta_b = \frac{3.5\sigma}{N^{1/3}}, \tag{4.78}$$

where σ is the sample standard deviation, and N is the sample size. This rule asymptotically minimizes the mean integrated square error (see eq. 4.14) and assumes that the underlying distribution is Gaussian; see [22]. An attempt to generalize this rule to non-Gaussian distributions is the Freedman–Diaconis rule,

$$\Delta_b = \frac{2(q_{75} - q_{25})}{N^{1/3}} = \frac{2.7\sigma_G}{N^{1/3}}, \tag{4.79}$$

which estimates the scale ("spread") of the distribution from its interquartile range (see [12]). In the case of a Gaussian distribution, Scott's bin width is 30% larger than the Freedman–Diaconis bin width. Some rules use the extremes of observed values to estimate the scale of the distribution, which is clearly inferior to using the interquartile range when outliers are present.

Although the Freedman–Diaconis rule attempts to account for non-Gaussian distributions, it is too simple to distinguish, for example, multimodal and unimodal distributions that have the same σ_G

Knuth's rule for estimating bin width

Knuth shows that the best piecewise constant model has the number of bins, M, which maximizes the following function (up to an additive constant, this is the logarithm of the posterior probability):

data

$$F(M|\{x_i\},I)) = N\,\log M + \log\left[\Gamma\left(\frac{M}{2}\right)\right] - M\log\left[\Gamma\left(\frac{1}{2}\right)\right] - \log\left[\Gamma\left(N+\frac{M}{2}\right)\right] + \sum_{k=1}^{M}\log\left[\Gamma\left(n_k+\frac{1}{2}\right)\right], \quad (5.107)$$

This is Bayesian model selection of M models

where Γ is the gamma function, and n_k is the number of measurements x_i, $i = 1,\ldots,N$, which are found in bin k, $k = 1,\ldots,M$. Although this expression is more involved than the "rules of thumb" listed in §4.8.1, it can be easily evaluated for an *arbitrary* data set.

Knuth derived eq. 5.107 using Bayesian model selection and treating the histogram as a piecewise constant model of the underlying density function. By assumption, the bin width is constant and the number of bins is the result of model selection. Given the number of bins, M, the model for the underlying pdf is

$$h(x) = \sum_{k=1}^{M} h_k \Pi(x|x_{k-1},x_k), \quad (5.108)$$

where the boxcar function $\Pi = 1$ if $x_{k-1} < x \le x_k$, and 0 otherwise. The M model parameters, h_k, $k = 1,\ldots,M$, are subject to normalization constraints, so that there are only $M-1$ free parameters. The uninformative prior distribution for $\{h_k\}$ is given by

$$p(\{h_k\}|M,I) = \frac{\Gamma(\frac{M}{2})}{\Gamma(\frac{1}{2})^M}\left[h_1\,h_2\ldots h_{M-1}\left(1-\sum_{k=1}^{M-1}h_k\right)\right]^{-1/2}, \quad (5.109)$$

which is known as the Jeffreys prior for the multinomial likelihood. The joint data likelihood is a multinomial distribution (see §3.3.3)

$$p(\{x_i\}|\{h_k\},M,I) \propto h_1^{n_1}\,h_2^{n_2}\ldots h_M^{n_M}. \quad (5.110)$$

Scargle's Bayesian Blocks algorithm

- Knuth's method assumes constant bin width!
- We can use the same Bayesian machinery to generalize Knuth's model to varying bin width (these are additional model parameters, just like the other ones)

In the Bayesian blocks formalism, the data are segmented into *blocks*, with the borders between two blocks being set by *changepoints*. Using a Bayesian analysis based on Poissonian statistics within each block, an objective function, called the log-likelihood fitness function, can be defined for each block:

$$F(N_i, T_i) = N_i(\log N_i - \log T_i), \tag{5.113}$$

where N_i is the number of points in block i, and T_i is the width of block i (or the duration, in time-series analysis). Because of the additive nature of log-likelihoods, the fitness function for any set of blocks is simply the sum of the fitness functions for each individual block. This feature allows for the configuration space to be explored quickly using dynamic programming concepts: for more information see [31] or the Bayesian blocks implementation in AstroML.

[31] Scargle, J. D., J. P. Norris, B. Jackson, and J. Chiang (2012).
Studies in astronomical time series analysis. VI.
Bayesian block representations. ArXiv:astro-ph/1207.5578.

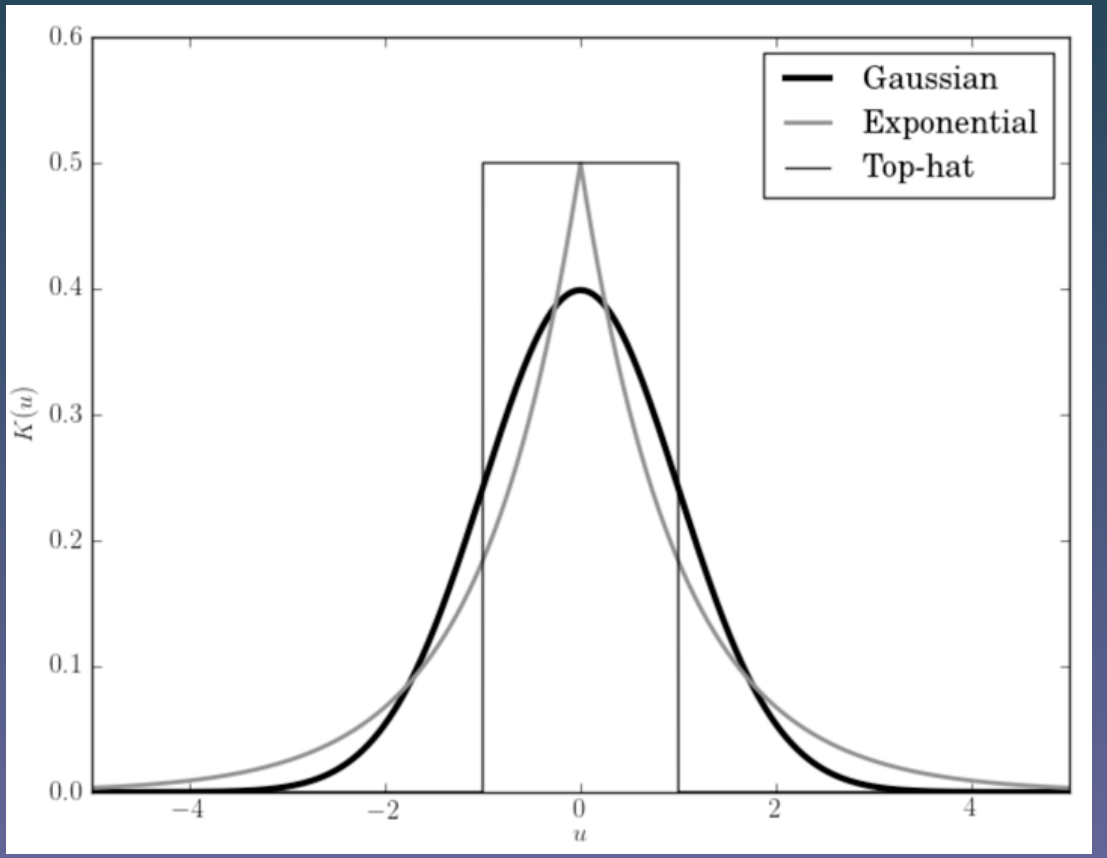
Kernel Density Estimates (KDE)

the underlying pdf is

$$h(x) = \sum_{k=1}^{M} h_k \Pi(x|x_{k-1}, x_k),$$

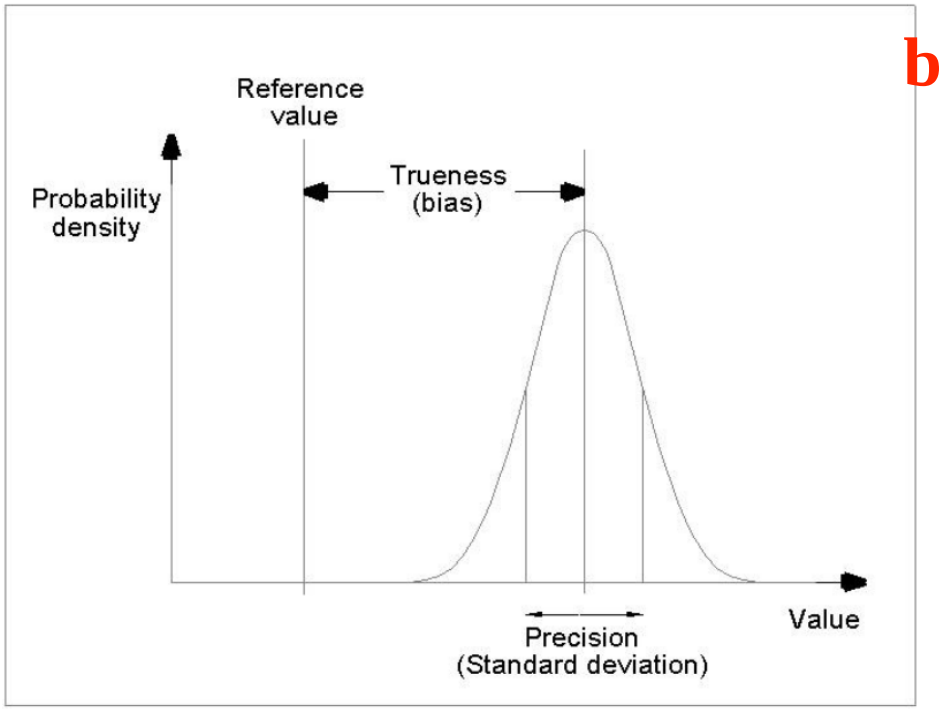
where the boxcar function $\Pi = 1$ if $x_{k-1} < x \leq x_k$, and 0 otherwise.

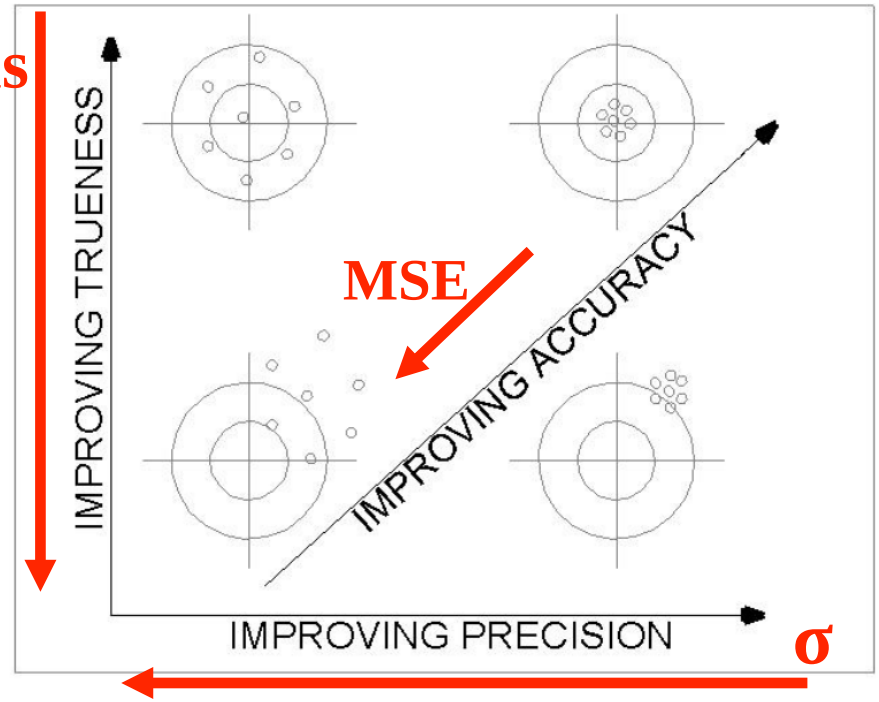
When computing histogram, we replace each object by the boxcar (top-hat) function. This function is called **kernel**. But of course we can use any other function (well, non-negative, normalized to 1, zero-mean, finite variance). This is the main idea of the KDE; it also works in high-D spaces.



• Measurements with known errors

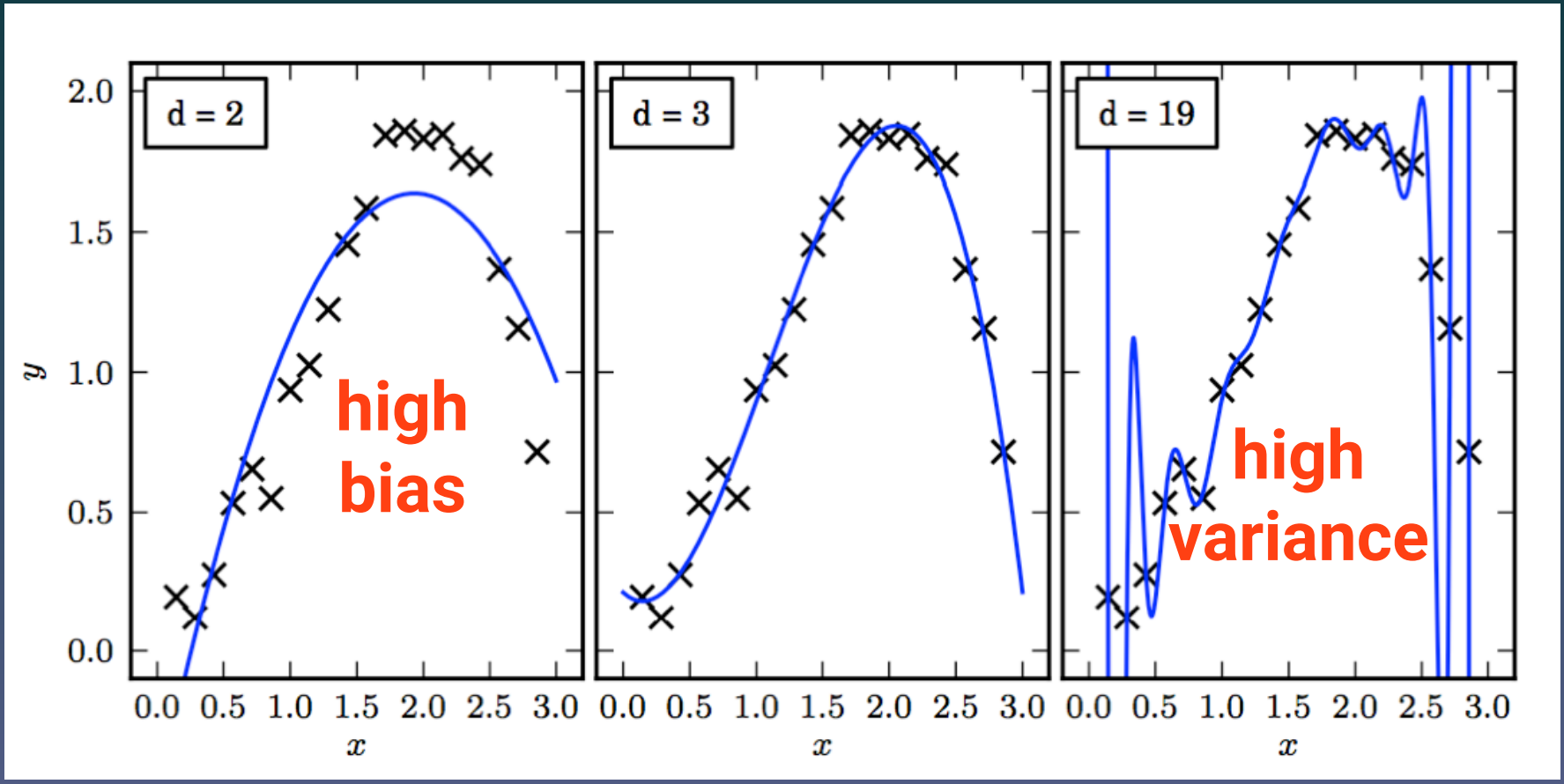
Let's assume that we have N measurements x_i, and that for each measurement we know the corresponding error distribution, that is, the expected distribution of x_i around the true value μ (which we want to estimate)





Mean Squared Error: $\mathrm{MSE} = V + \mathrm{bias}^2$ (V=variance=σ^2)

Bias vs. variance tradeoff



high bias: "can't fit" data points
high variance: "oscillations" between data points (how would we know? c.f. cross-validation method: "leave one out")

Extreme Deconvolution in high-D (XD)

Bovy, Hogg & Roweis 2011 (arXiv:0905.2979)

- **Two basic assumptions:**

  a) the sampled population distribution is a arbitrary mixture of high-D Gaussian components, and

  b) data have heteroscedastic errors with known covariance matrix

being α_i. Thus, the pdf of $\mathbf{x}$ is given as

$$p(\mathbf{x}) = \sum_j \alpha_j \mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)\,, \tag{6.18}$$

where, recalling eq. 3.97,

$$\mathcal{N}(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^D \det(\mathbf{\Sigma}_j)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \mathbf{\Sigma}_j^{-1}(\mathbf{x}-\mu)\right). \tag{6.19}$$

Extreme deconvolution generalizes the EM approach to a case with measurement errors. More explicitly, one assumes that the noisy observations $\mathbf{x}_i$ and the true values $\mathbf{v}_i$ are related through

$$\mathbf{x}_i = \mathbf{R}_i \mathbf{v}_i + \epsilon_i, \tag{6.20}$$

XD is a very powerful method: it can treat incomplete and heterogeneous data

Extreme Deconvolution in high-D (XD)

- make this plot by running
  %run fig_XD_example.py

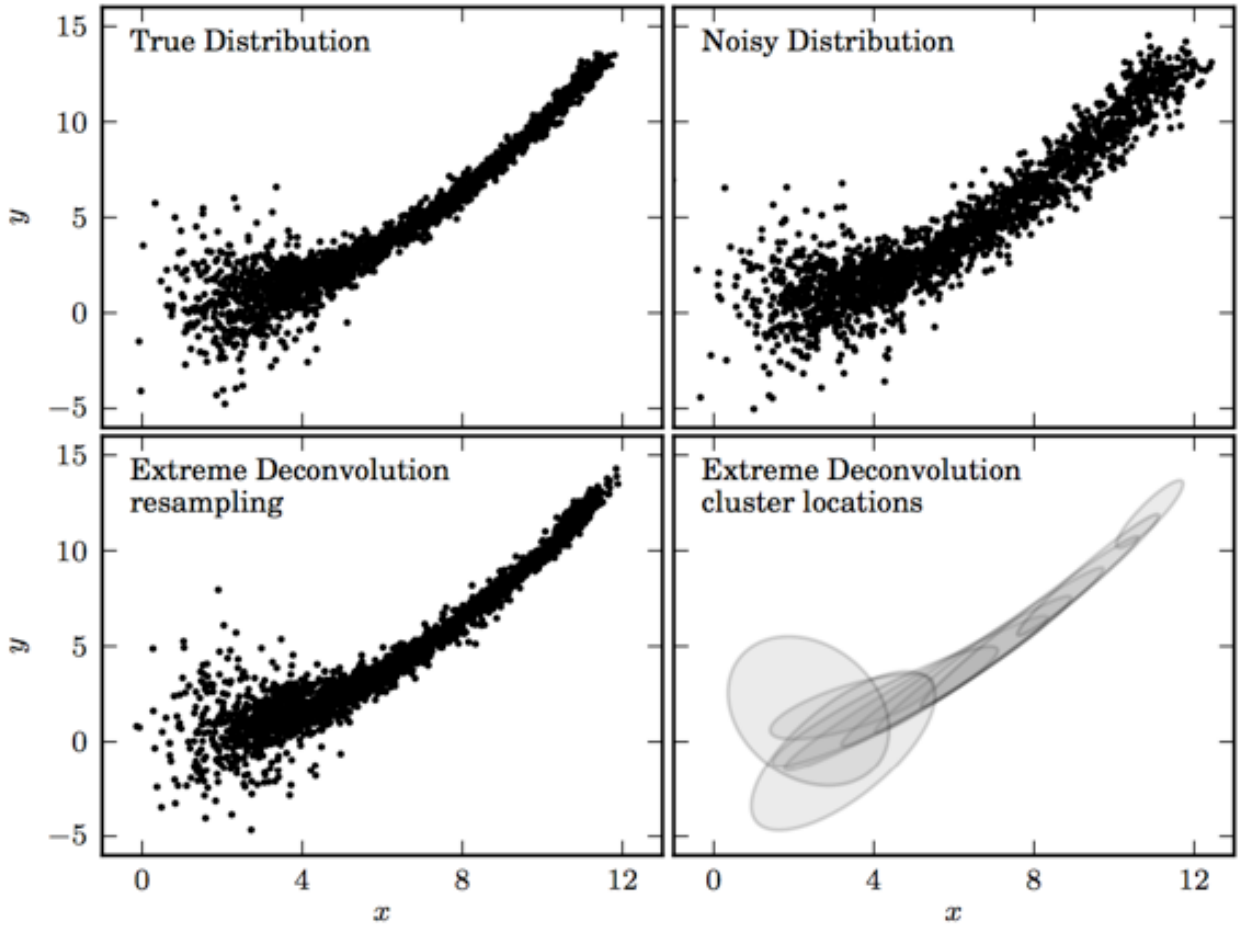
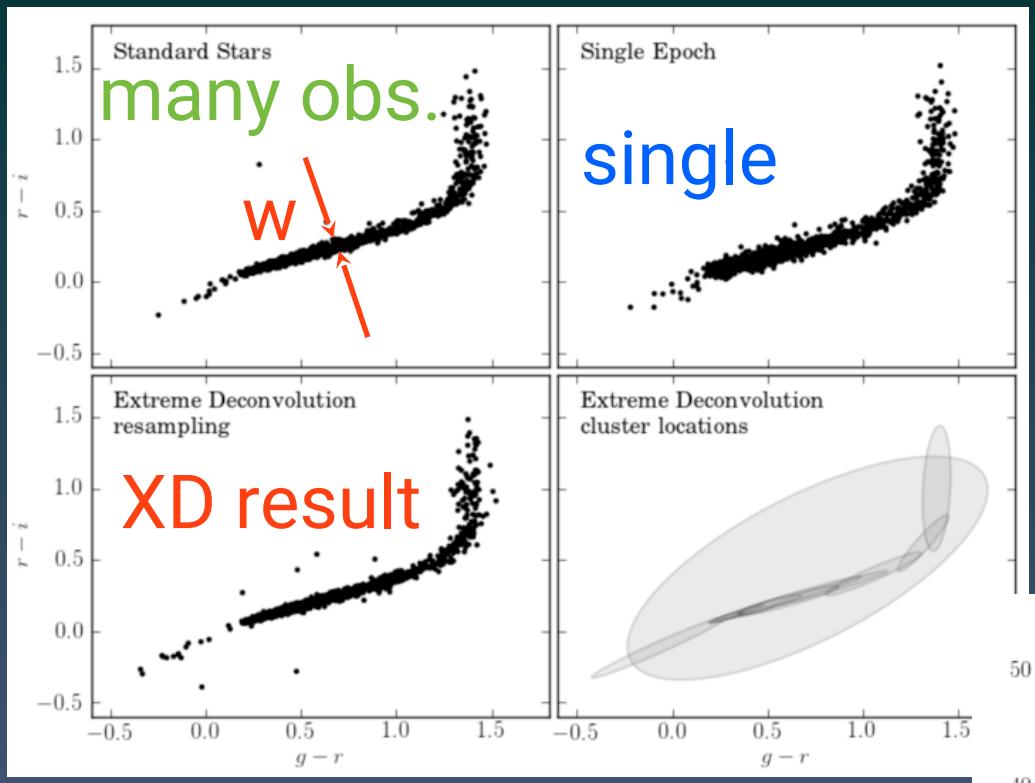
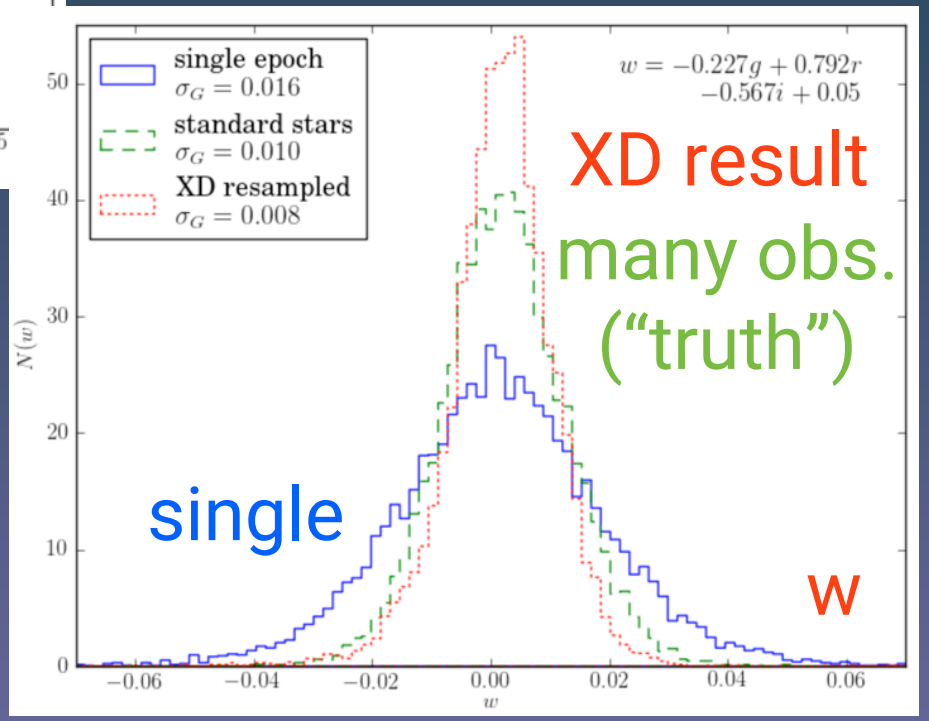


Figure 6.11.: An example of extreme deconvolution showing a simulated two-dimensional distribution of points, where the positions are subject to errors. The top two panels show the distributions with small (left) and large (right) errors. The bottom panels show the densities derived from the noisy sample (top-right panel) using extreme deconvolution; the resulting distribution closely matches that shown in the top-left panel.

Extreme Deconvolution in high-D (XD)



XD is a very powerful method: if GMM is able to model the observed distribution (usually OK)



XD result: the intrinsic width of the r-i vs. g-r stellar locus is ~0.01 mag., in agreement with high-SNR averaged multi-epoch photometry

W7 Dimensionality reduction and regression

- **Dimensionality Reduction**
  - Covariance and Correlation
  - **Principal Component Analysis**
  - Non-negative Matrix Factorization
  - Independent Component Analysis
  - Manifold learning (Locally Linear Embedding)
- Regression and a few misc. points
  - (Gaussian) **errors in both variables**
  - regression with **non-Gaussian errors and/or outliers**
  - (fast matching using KD trees)

Principal Component Analysis (PCA)

Motivation:

High-D case (driven by the "curse of dimensionality"): there are ~4000 data points in SDSS spectra. Can we represent these spectra as linear combinations of a **much smaller** number of eigen-components?

The curse of dimensionality:

in high-D space, the ratio of the volume of the unit hyper-sphere and the volume of the hyper-cube that encloses it goes to 0 with as D increases

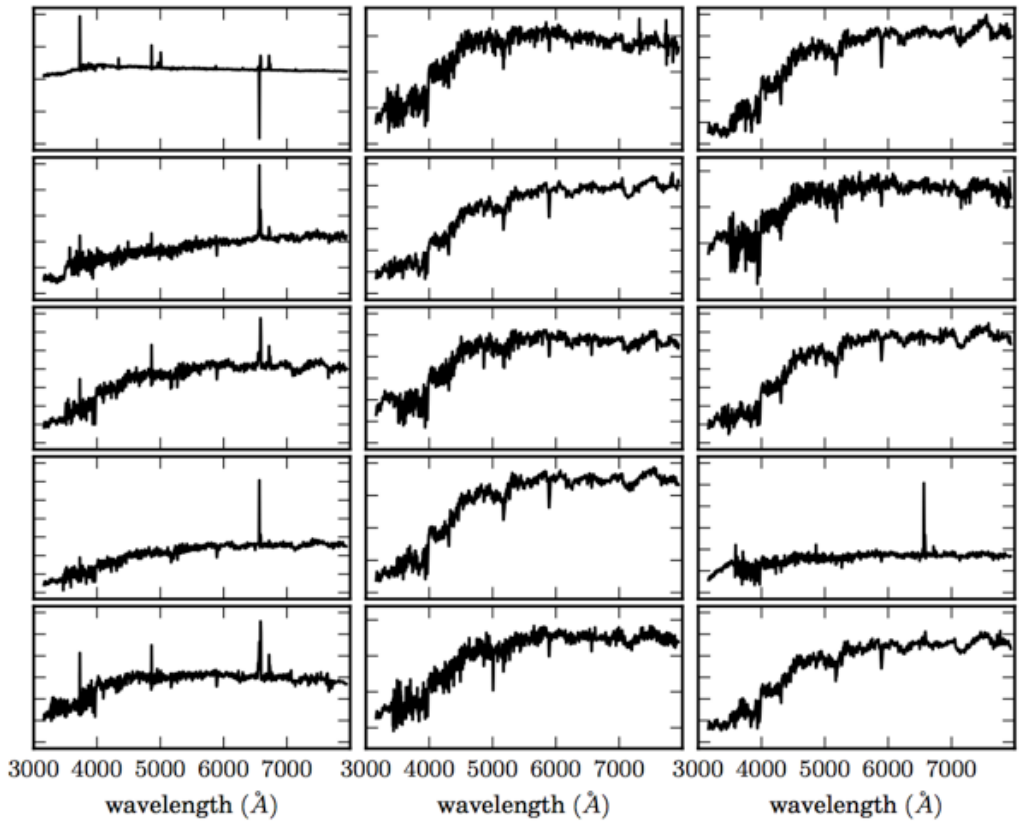


Figure 7.1.: A sample of 15 galaxy spectra selected from the SDSS spectroscopic data set (see §1.5.5). These spectra span a range of galaxy types, from star-forming to passive galaxies. Each spectrum has been shifted to its rest frame and covers the wavelength interval 3000–8000 Å. The specific fluxes, $F_\lambda(\lambda)$, on the ordinate axes have an arbitrary scaling.

Principal Component Analysis (PCA)

Each spectrum $\boldsymbol{x}_i(k)$ can be described by

$$x = \mu_x + P_1 \cos\alpha - P_2 \sin\alpha,$$
$$y = \mu_y + P_1 \sin\alpha + P_2 \cos\alpha.$$

$$\boldsymbol{x}_i(k) = \boldsymbol{\mu}(k) + \sum_j^R \theta_{ij} \boldsymbol{e}_j(k), \tag{7.17}$$

where i represents the number of the input spectrum, j represents the number of the eigenspectrum, and, for the case of a spectrum, k represents the wavelength. Here, $\boldsymbol{\mu}(k)$ is the mean spectrum and θ_{ij} are the linear expansion coefficients derived from

$$\theta_{ij} = \sum_k \boldsymbol{e}_j(k)(\boldsymbol{x}_i(k) - \boldsymbol{\mu}(k)). \tag{7.18}$$

R is the total number of eigenvectors (given by the rank of X, $\min(N,K)$). If the summation is over all eigenvectors, the input spectrum is fully described with no loss of information. Truncating this expansion (i.e., $r < R$),

$$\boldsymbol{x}_i(k) = \sum_i^{r<R} \theta_i \boldsymbol{e}_i(k), \tag{7.19}$$

will exclude those eigencomponents with smaller eigenvalues. These components will, predominantly, reflect the noise within the data set. This is reflected in figure 7.5: truncating the recon-

Principal Component Analysis (PCA)

sometimes called Karhunen-Loeve and Hotelling transform

How is it done:

Data matrix is formed by N sets (objects) of K measured features (attributes); e.g. K~4000 for SDSS spectra of, say, N=100,000 quasars; we subtract the mean of each feature (and sometimes normalized by their st. deviation, called whitening; in case of spectra, flux is normalized to 1 to avoid uninteresting correlations with source brightness)

Using matrix algebra, the first eigen-component is found by maximizing variance and other eigen-components by requiring covariance to vanish.

There are different approaches, whose performance depends on N and K.

Principal Component Analysis (PCA)

sometimes called Karhunen-Loeve and Hotelling transform

How is it done:

Different matrix algebra approaches to PCA:

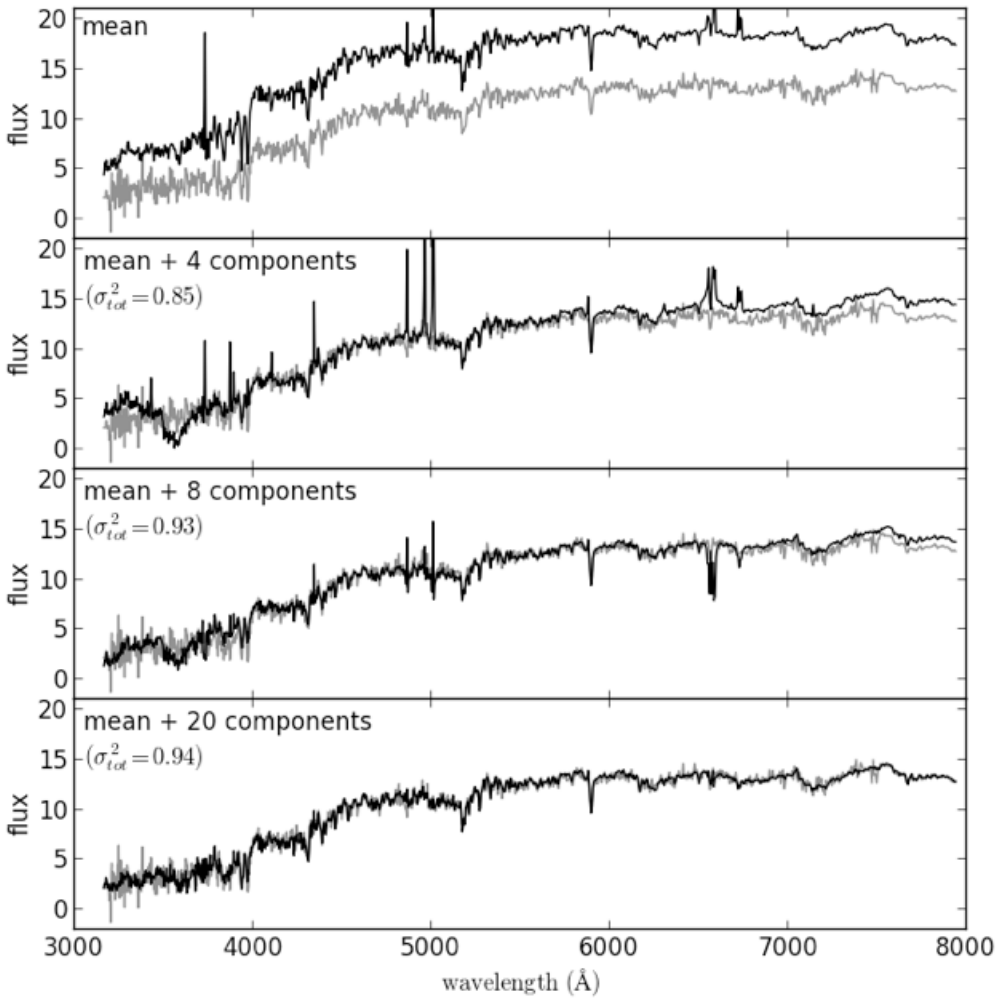
1) Singular Value Decomposition of the data matrix:
efficient in most practical cases (exceptions below)

2) Eigenvalue Decomposition of the covariance matrix:
efficient for N >> K

3) Eigenvalue Decomposition of the correlation matrix:
efficient for K >> N

Principal Component Analysis (PCA)

- make this plot by running
  %run fig_spec_reconstruction.py



How to choose the number of eigencomponents ?

Mean spectrum

4 eigencomponents

8 eigencomponents

20 eigencomponents

Principal Component Analysis (PCA)

How to choose the number of eigencomponents?

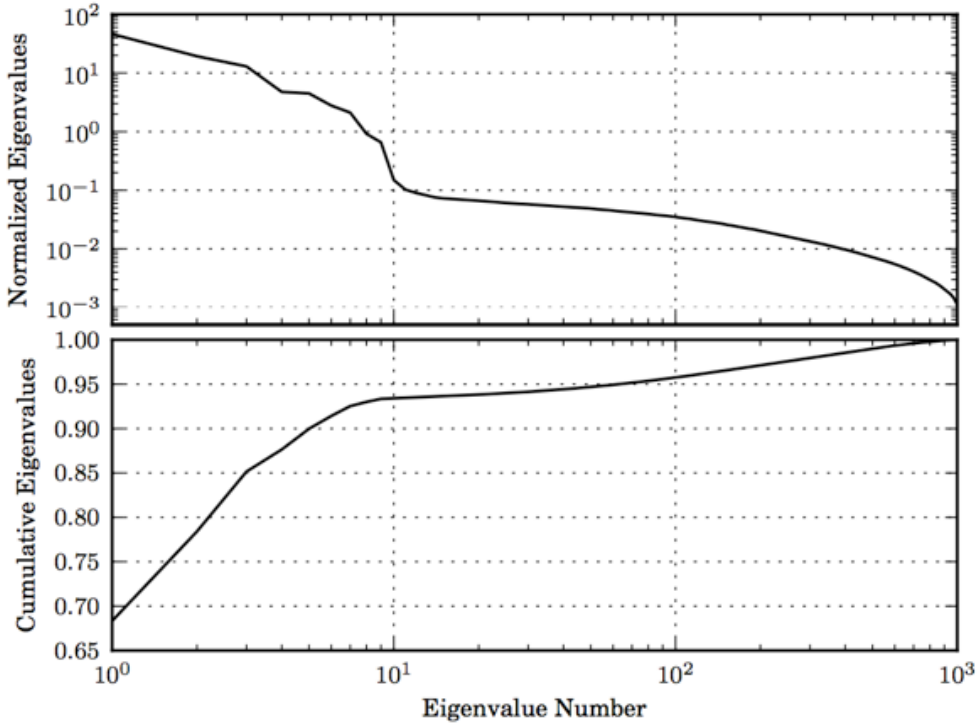


Figure 7.4.: The eigenvalues for the PCA decomposition of the SDSS spectra described in §7.3.2. The top panel shows the decrease in eigenvalue as a function of the number of eigenvectors, with a break in the distribution at ten eigenvectors. The lower panel shows the cumulative sum of eigenvalues normalized to unity. 94% of the variance in the SDSS spectra can be captured using the first ten eigenvectors.

There is no universal method, but typically we require that some fraction of data variance is captured by truncated series.

This plot (called the scree plot) shows that the first ten eigencomponents already capture about 94% of the variance in the dataset. Higher terms attempt to describe high-frequency measurement noise.

Principal Component Analysis (PCA)

A few more points about PCA

It can treat missing data by introducing weights for each point (Connolly & Szalay, 1999, AJ 117, 2052).

If the dataset cannot fit in memory, then PCA computation can be challenging. One way to address this is the use of iterative online algorithms.

In some problems, linear decomposition might not be appropriate (e.g. a set of Planck functions with varying temperature)

Eigencomponents can be negative, which in some applications provides only limited insight into underlying physics (e.g. galaxy spectra).

Glossary

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant= $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

From: http://statweb.stanford.edu/~tibs/stat315a/glossary.pdf

Which dimensionality reduction technique to use in practice?

Simple summary. Table 7.1 is a simple summary of the trade-offs along our axes of accuracy, interpretability, simplicity, and speed in dimension reduction methods, expressed in terms of high (H), medium (M), and low (L) categories.

Table 7.1.: Summary of the practical properties of the main dimensionality reduction techniques.

Method	Accuracy	Interpretability	Simplicity	Speed
Principal component analysis	H	H	H	H
Locally linear embedding	H	M	H	M
Nonnegative matrix factorization	H	H	M	M
Independent component analysis	M	M	L	L